Computer Science 350

Assignment Two

Steven Kerr 6022796

11/05/2019

1. i

A decider is a machine that halts on all inputs. For a proof that M is not a decider we must first find a string that, when passed to machine M, will not halt.
Let $u = 01$ and let us run u on M.

$\vdash_M q_1 01 \sqcup \sqcup \sqcup \ldots$
$\vdash_M 0 q_2 1 \sqcup \sqcup \sqcup \ldots$
$\vdash_M 01 q_3 \sqcup \sqcup \sqcup \ldots$
$\vdash_M 01 \sqcup q_3 \sqcup \sqcup \ldots$
$\vdash_M 01 \sqcup \sqcup q_3 \sqcup \ldots$

as we can see here, when q_3 is reached, the input can never reach a final state ie:
$\delta(q_3, \sqcup) = (q_3, \sqcup, R)$
$\delta(q_3, 0) = (q_3, 0, R)$
$\delta(q_3, 1) = (q_3, 1, R)$
Therefore machine M is not a decider.

1. ii

The language recognised by a Turing-machine is, by definition, the set of strings that the machine accepts. Let such a set be called L and let
$L = \{1^n 0^k | n \geq 0 \text{ and } k > 0\}$

Now we must prove $A = L(M)$
By definition, if M is a turing machine with input alphabet Σ, let $L(M)$ be the set of strings over Σ accepted by M. If A is a language over Σ, we say that M recognises L if $A = L(M)$.
So, I must prove that $A \subseteq L(M)$ and $L(M) \subseteq A$.

$i) A \subseteq L(M)$
Suppose that $w \in A$. So w contains $n\#1^s$ and $k\#0^s$ where $n \geq 0$ and $k > 0$. Let $w = w_1 w_2 \dots w_n$ and let w_k be the first 0 in w ($1 \leq k \leq n$). so $w = 1^{k-1}0^j$ and $j = n - (k-1)$.
We need to show that $w \in L(M)$, ie that M accepts w. So we need to show that there is a sequence of configurations $C_1 \dots, C_|$ such that C_1 is the starting configuration of M on w, each C_i yields $_M C_{i+1}$ (where $1 \leq i < n$), and $C_|$ is an accepting configuration.
We construct the following sequence of cfs on M:
$C_1 = q_1 1^{k-1} 0^j \sqcup \sqcup \dots,$
$C_2 = 1^1 q_1 1^{k-2} 0^j \sqcup \sqcup \dots,$
$C_k = 1^{k-1} q_1 0^j \sqcup \sqcup \dots\ ,$
$C_{k+1} = 1^{k-1} 0 q_2 0^{j-1} \sqcup \sqcup \dots,$
$C_{k+2} = 1^{k-1} 00 q_2 0^{j-2} \sqcup \sqcup \dots,$
$C_{n-1} = 1^{k-1} 0^{j-1} q_2 0 \sqcup \sqcup \dots,$
$C_n = 1^{k-1} 0^j q_2 \sqcup \sqcup \dots,$
$C_{n+1} = 1^{k-1} 0^j \sqcup q_{accept} \sqcup \dots$

It is obvious that $C_i \vdash_M C_{i+1}$ for $1 \leq i < k$, since $\delta(q_1, 1) = (q_1, 1, R)$ and so until the machine encounters the first 0, ie w_k, it simply stays in q_1 and moves right. In addition, $C_k \vdash_M C_{k+1}$, since $\delta(q_2, 0) = (q_2, 0, R)$ and so until the machine encounters the first $\sqcup$, ie w_{n+1}, it simply stays in q_2 and moves to the right. Finally, $C_n \vdash_M C_{n+1}$ [Reason: $C_n = 1^{k-1} 0^j q_2 \sqcup \sqcup \dots,$ and this yields $_M\ C_{n+1} = 1^{k-1} 0^j \sqcup q_{accept} \sqcup \dots$ since $\delta(q_2, \sqcup) = (q_{accept}, \sqcup, R)$]

Note also that C_1 is the start configuration of M on w, and C_{n+1} is an accepting configuration. It follows from definitions 3 M accepts w. Hence since w was an arbitrary member of A, it follows that for all $w \in 0, 1^*$, if $w \in A$ then $w \in L(M)$. Hence $(^*) A \subseteq L(M)$.

$ii)L(M) \subseteq A$

Consider $w \in L(M)$. Since $w \in L(M)$, M accepts w. So there is a sequence of cfs $C_1, C_2, \ldots, C_{n+1}$ such that C_1 is the start cf of M on w, each $C_i \vdash_M C_{i+1} (for 0 \leq i \leq k)$, also $C_k \vdash_M C_{k+1} (for 0 \leq k < n)$, and C_{n+1} is an accepting cf.

Suppose that $C_{n+1} = 1^k 0^j q_{accept}$. Clearly $w = 1^k 0^j$ where $k \geq 0$ and $j > 0$ (since M doesn't change any of the input symbols). Now note that the application of δ that yields C_{n+1} from the preceding cf C_n is $\delta(q_2, \sqcup) = (q_{accept}, \sqcup, R)$ and to get to q_2 there must be atleast one 0 in w as from C_k to C_{k+1} is $\delta(q_1, 0) = (q_2, 0, R)$. It follows that w contains one or more 0^s and zero or more 1^s, and so $w \in A$.

Since w was an arbitrart member of $L(M)$, it follows that for all $w \in \{0,1\}^*$, if $w \in L(M)$ then $w \in A$. Hence $L(M) \subseteq A$.

1. iii

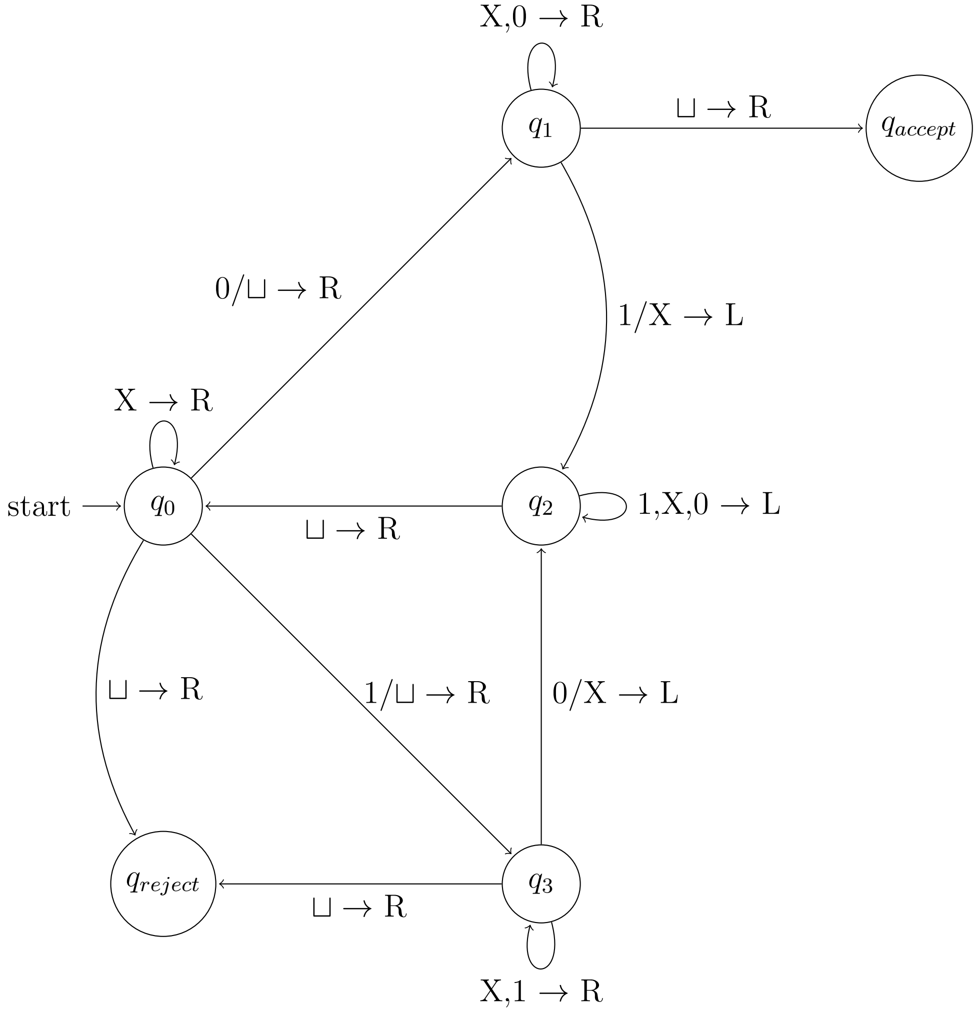
The language A is turing-decidable. This is because there exists a turing-machine that decides A. As a simple example, if we took the turing-machine Q, removed q_3 and redirected anything going to q_3 to q_{reject} and called it Q_2 we would have:

$\delta(q_1, \sqcup) = (q_{reject}, \sqcup, R)$
$\delta(q_1, 0) = (q_2, 0, R)$
$\delta(q_1, 1) = (q_1, 1, R)$
$\delta(q_2, \sqcup) = (q_{accept}, \sqcup, R)$
$\delta(q_2, 0) = (q_2, 0, R)$
$\delta(q_2, 1) = (q_{reject}, 1, R)$

As you can see here, Q_2 decides language A. In q_1 the input can be rejected, looped on 1^s or sent to q_2. q_2 can be accepted, loop on 0^s or rejected. Therefore , Q_2 decides A and the language A is decidable.

2. i

Turing-Machine M



A formal definition of the above Turing machine

- $M = (Q,\ \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject})$
- $Q = (q_0, q_1, q_2, q_3, q_{accept}, q_{reject})$
- $\Sigma = \{0,\ 1\}^*$
- $\Gamma = \{0,\ 1,\ \sqcup,\ X\}$

δ	0	1	$\sqcup$	X
q_0	q_1	q_3	q_{reject}	q_0
q_1	q_1	q_2	q_{accept}	q_1
q_2	q_2	q_2	q_0	q_2
q_3	q_2	q_3	q_{reject}	q_3

description:

This machine matches 1^s and 0^s together.

If there is no match for a 0, then there are
more 0^s than 1^s.
When the machine is started q_0 checks the input. If the input is 0 then the machine replaces 0 with $\sqcup$ and moves right to q_1. q_1 then moves right through all the X^s and 0^s looking for a 1. Once a 1 has been located (this means that a 0 has been matched with a 1), the machine replaces 1 with an X and then moves left into q_2.
In q_2 the machine moves left through all the 1^s, X^s and 0^s looking for $\sqcup$ (the initial matched item). Once $\sqcup$ has been found the machine returns to q_0 to start again. Note that q_0 moves right over all the X^s looking for the first instance of 1 or 0.
If the input is 1 then the machine replaces 1 with $\sqcup$ moves to q_3. q_3 then moves right through all the X^s and 1^s looking for a 0. Once a 0 has been located (this means that a 1 has been matched with a 0), the machine replaces 0 with an X and then moves left into q_2 (q_2 has already been explained).

The machine moves right into q_{accept} if a $\sqcup$ is found in q_1. This means that no matching 1 was found for a 0 which then means that there are more 0^s than 1^s.

The machine moves right into q_{reject} in two cases:
1. If the string is the empty string. This means that there are not more 0^s than 1^s and is therefore rejected.
2. If the machine is in q_3 and a $\sqcup$ is found. This means that no matching 0 was found for a 1 which then means that there are more 1^s than 0^s.

2. ii

Computation for machine M on string 01:
$\vdash_M q_0 01 \sqcup \sqcup \ldots$
$\vdash_M \sqcup q_1 1 \sqcup \sqcup \ldots$
$\vdash_M q_2 \sqcup X \sqcup \sqcup \ldots$
$\vdash_M \sqcup q_0 X \sqcup \sqcup \ldots$
$\vdash_M \sqcup X q_0 \sqcup \sqcup \ldots$
$\vdash_M \sqcup X \sqcup q_{reject} \sqcup \ldots$
Computation for machine M on string 100:
$\vdash_M q_0 100 \sqcup \sqcup \ldots$
$\vdash_M \sqcup q_3 00 \sqcup \sqcup \ldots$
$\vdash_M q_2 \sqcup X0 \sqcup \sqcup \ldots$
$\vdash_M \sqcup q_0 X0 \sqcup \sqcup \ldots$
$\vdash_M \sqcup X q_0 0 \sqcup \sqcup \ldots$
$\vdash_M \sqcup X \sqcup q_1 \sqcup \sqcup \ldots$
$\vdash_M \sqcup X \sqcup \sqcup q_{accept} \sqcup \ldots$

2. iii

We start by assuming that there is a turing machine N that decides L_1. We can then construct a turing machine N' that decides $\overline{L_1}$ in the following way:
N' = "On input w":

1. Simulate N on w.
2. *Accept* if N rejects, *reject* if N accepts.

N' always halts because N always halts. N' gives the correct result, because if $w \in L_1$ N will accept and so N' will reject, and if $w \notin L_1$, N will reject in which case N' will accept.
This means that N' decides $\overline{L_1}$ and the language is decidable.
We now have machine N' that decides $\overline{L_1}$ and we have machine M that decides the language L. Using these two machines we can now construct a turing machine M' that decides L_2 in the following way:
M' = "On input w":

1. Simulate N' on w.
2. if N' accepts, move to 3. if N' rejects, *reject*.
3. Simulate M on w.
4. if M accepts, *reject*. if M rejects, *accept*

M' always halts because N' and M always halts. M' gives the correct result, because if $w \notin \overline{L_1}$ N' will reject and so if $w \in \overline{L_1}$, then w will be sent to M. In M if $w \notin L$, M will accept w, else M will reject.
This means that M' decides L_2 and the language is decidable.

3.

$Q = \{< A, B > \mid$ A and B are NFAs over some alphabet Σ and $\emptyset \sqsubseteq L(A)$ and $L(A) \sqsubseteq L(B)$ is Turing-decidable.

Note $\sqsubseteq$ = 'Is a proper set of'

Proof: Firstly, we know that any NFA can be replicated by a DFA. Secondly, by definition, we know the empty set is a proper subset of any nonempty set. So, we know that $L(A)$ is not an empty set.

Again, by definition, we know that $L(A) \sqsubseteq L(B)$ iff:

$\forall x \in L(A), x \in L(B)$ but $\forall y \in L(B), \neg(\forall y) \in L(A)$ which implies that:

$L(A) \cap \overline{L(B)} = \emptyset \cap L(B) \cap \overline{L(A)} \neq \emptyset$

From the proofs of the closure of the class of regular languages under intersection, complement and union, we can construct a DFA C which recognises $L(A) \cap \overline{L(B)} = \emptyset \cap L(B) \cap \overline{L(A)} \neq \emptyset$.

This construction can be carried out by a TM. Once we have such a C, we can test to see if $L(A) \sqsubseteq L(B)\}$ by testing to see if $L(A) \cap \overline{L(B)} = \emptyset$ and $L(B) \cap \overline{L(A)} \neq \emptyset$ using the TM C^*.

C^* = 'On input $< A$,$B >$, where A and B are NFAs:',

1. Convert A, B to DFAs A', B'.
2. Use T on $< A' >$ from 'Theorem 10' to show that $L(A)$ is not empty. If T rejects then got to 3, else *reject*.
3. let B'' be the DFA that recognises $\overline{L(B)}$ let D be a DFA that recognises $L(A) \cap \overline{L(B)}$. Use T on $< D >$, If T rejects then *reject*, else got to 4.
4. let A'' be the DFA that recognises $\overline{L(A)}$ let E be the DFA that recognises $L(B) \cap \overline{L(A)}$. Use T on $< E >$, If T accepts then *reject*, if T rejects then *accept*.

Clearly C^* accepts $< A, B >$ iff $L(A) \sqsubseteq L(B)$. C^* will always end in a *reject* or *accept* state because we make use of the theorem 10 E_{DFA} is decidable. In step 1 we convert the NFAs to DFAs. Then in step 2 we check to see if $L(A)$ is empty. If $L(A) \neq \emptyset$ then we move to step 3 and check to see if $L(A) \cap \overline{L(B)} = \emptyset$ if accepted then we move to step 4 and check $L(B) \cap \overline{L(A)} \neq \emptyset$. As we can see, C^* will always halt on input $< A, B >$ because of theorem 10 where a E_{DFA} is decidable.

4.

$\odot(A,B) = \{w \in \{0,1\}^* | (w \in A \;\&\; w \in B) \; or \; w \in A \circ B\}$
$\otimes(A,B) = \{w \in \{0,1\}^* | (w \in A \;\&\; w \notin B) \; or \; w \in A \circ B\}$
Note, the symbols $\odot$ and $\otimes$ will represent given language. We can also assume that languages A and B are decidable.
Proof: I will start with deciding $A \circ B$. (Theorem 3. iv).
Suppose A and B are Turing-decidable. Let M_1 decide A and M_2 decide B. Then N, a NTM, decides $A \circ B$. N is as follows:

$N =$ 'On input w',

1. Non-deterministically split w into two parts, x and y. Go to 2.
2. Run M_1 on the left part, x. If M_1 accepts, go to 3. if M_1 rejects, go to q_{reject}.
3. Run M_2 on the right part y. If M_2 accepts, *accept*. If M_2 rejects, go to q_{reject}.

Clearly N accepts w iff w is the concatenation of two strings x and y, the first of which is in A and the second in B, and N rejects otherwise. Hence N decides $A \circ B$.

Now we will look at $(w \in A \;\&\; w \notin B)$. This means that $A \cap B = \emptyset$. Also, $(w \in A \;\&\; w \in B)$ would then mean that $A \cap \overline{B} = \emptyset$ and we can use this to say that:

$$\bigodot = A \cap \overline{B} = \emptyset$$

$$\bigotimes = A \cap B = \emptyset$$

We can use the following two Turing machine's M^* to show that the set of Turing decidable languages are closed under $\odot$ and M^{**} to show that the set of Turing decidable languages are closed under $\otimes$:

Let A' be the DFA that accepts the language A. Let B' be the DFA that accepts the language B and run both M^* and M^{**} in the following way:

M^* = 'On input A and B, where A and B are decidable languages'

1. Let B'' be the DFA that decides $\overline{B}$ and let D be the DFA that recognises $(A \cap \overline{B})$. Use T (from theorem 10) on $< D >$, if T accepts, *accept*. If T rejects got to 2.

2. Take a string w such that $w \in \{0, 1\}^*$ and use Turing machine N on w. If N accepts, *accept*. If N rejects, *reject*.

M^{**} = 'On input A and B, where A and B are decidable languages'

1. Let D^* be the DFA that recognises $(A \cap B)$. Use T on $< D^* >$. If T accepts *accept*. If T rejects, go to 2.

2. Take a string w such that $w \in \{0, 1\}^*$ and use Turing machine N on w. If N accepts, *accept*. If N rejects, *reject*.

As we can see, machine M^* and M^{**} can only *accept* or *reject* the two languages A and B. This is because we use T from theorem 10 which decides a DFA input. So T will always halt in an accept or reject state. We also make use of Theorem 3. iv where Turing machine N decides concatenation of two decidable languages. N is also decidable because N will always halt in an accept or reject state. We already know the languages A and B are Turing-decidable and we can see that M^* and M^{**} are also decidable. Therefore, M^* decides:

$$\bigodot(A, B) = \{w \in \{0, 1\}^* | (w \in A \;\&\; w \in B) \textit{ or } w \in A \circ B\}$$

and M^{**} decides :

$$\bigotimes(A, B) = \{w \in \{0, 1\}^* | (w \in A \;\&\; w \notin B) \textit{ or } w \in A \circ B\}$$

The set of Turing-decidable languages is also closed under both $\bigotimes$ and $\bigodot$. We know this because we have proven above that both languages are decidable using Turing machines M^* and M^{**}. This means, by definition of closure The class of Turing-decidable languages over a fixed alphabet is closed under the operations of i) union, ii) intersection, iii) complement, iv) concatenation, and v) star. Therefore, we can perform all operations on $\bigotimes$ and $\bigodot$.

5. [bonus question]
Consider $L = \{1^n |$there is a consecutive run of at least n 7's in the decimal expansion of $\pi\}$. Prove that L is Turing-decidable.
We know that the decimal expansion of π is countably infinite. We can easily map the natural numbers N to $decimal^{\pi}$. This means that the decimal expansion of π is regular which means there is a DFA that accepts it.
It seems like this is possible but i am not sure about putting a limit on n. For instance,
Case 1: there is a consecutive run of atleast n 7's
Case 2: the machine never halts because n is too high.
If I put a limit on n then i know it will always halt. The question does state that 'atleast n 7's' which makes me think that we can put a bound on n.
Another thing i know about the first million digits of π:

1. '7' appears 10,287 times
2. '77' appears 943 times
3. '777' appears 93 times
4. '7777' appears 16 times
5. '77777' appears 0 times

Because we know that the the decimal expansion of π is countably infinite there is a possibility off the string 7^k where $k \geq 5$ in the next million digits of π so i propose two algorithms we can consider:

1. For every integer $n \geq 0$, the string 7^n appears in the decimal representation of π. In this case, the algorithim always returns *accept*.
2. There is a largest integer N such that 7^N appears in the decimal representation of π. In this case the following algorithm (with the value N hard-coded) is always correct:

We have no idea which of these algorithms is correct, or what value of N is the right one in algorithm 2. We do know that one of these algorithms is guaranteed to be correct. Therefore, there is an algorithm to decide whether a string of n 7^s appears in π. This makes L Turing-decidable.